(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 3 312 722 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
25.04.2018 Bulletin 2018/17

(51) Int Cl.:
G06F 9/48 *(2006.01)* ***G06F 9/50*** *(2006.01)*

(21) Application number: **17195347.4**

(22) Date of filing: **06.10.2017**

(84) Designated Contracting States:
AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR
Designated Extension States:
BA ME
Designated Validation States:
MA MD

(30) Priority: **21.10.2016 DE 102016220777**

(71) Applicant: **FUJITSU LIMITED**
Kanagawa 211-8588 (JP)

(72) Inventors:
- **MORA LÓPEZ, José**
  28017 Madrid (ES)
- **VILLAZÓN-TERRAZAS, Boris**
  28003 Madrid (ES)
- **DE LA TORRE, Victor**
  28007 Madrid (ES)
- **LLAVES, Alejandro**
  28011 Madrid (ES)
- **PEÑA MUÑOZ, Manuel**
  41006 Sevilla (ES)

(74) Representative: **Haseltine Lake LLP**
Lincoln House, 5th Floor
300 High Holborn
London WC1V 7JH (GB)

(54) DATA PROCESSING APPARATUS, METHOD, AND PROGRAM

(57) Embodiments include a data processing apparatus, the apparatus comprising: a software library, storing a plurality of software services, each executable software service being configured to execute a respective data processing function; a user interface configured to receive a plurality of user input commands, each user input command expressed in a domain specific language and defining a data processing target and a data processing request; and a parser. The parser is configured to extract from each user input command: the data processing request from the domain specific language; and the defined data processing target. The apparatus further comprises: a knowledge base, configured to maintain a record of the data processing request and the defined data processing target for each of the plurality of user input commands; a software service execution scheduler, configured, for each user input command, to obtain the data processing request from the parser, and to compile a schedule of one or more software services from among the plurality of software services to fulfil the data processing request; a software service execution controller configured, for each user input command, to control execution of the compiled schedule of one or more software services, the defined data processing target being the input data to the controlled execution, and to output a processing result of said controlled execution; and a result processor, configured to obtain the output processing result, and, based on the records of data processing requests and defined data processing targets maintained by the knowledge base, to identify a data processing request candidate for performance on the processing result, and to output to the user as a selectable user input command expressed in the domain specific language, via the user interface, the identified data processing request candidate with the processing result defined as a data processing target.

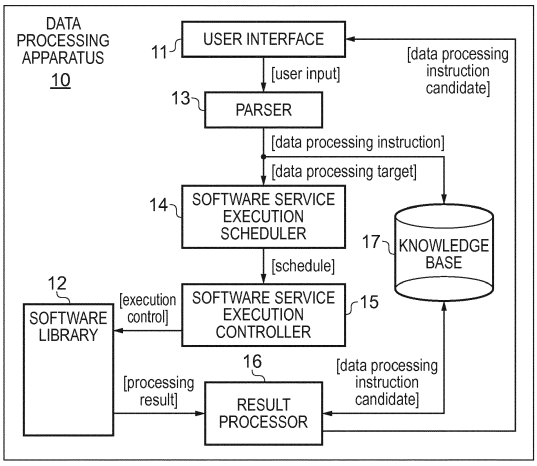

FIG. 1

EP 3 312 722 A1

Description

[0001] Embodiments lie in the field of data processing and in particular relate to the execution of software services via a user interface.

[0002] Data science is an interdisciplinary field about processes and systems to extract knowledge or insights from data in various forms, either structured or unstructured, which is a continuation of some data analysis fields such as statistics, data mining, and predictive analysis.

[0003] According to the New York Times, 80% of a typical data science project is sourcing, cleaning and preparing the data, while the remaining 20% is the actual data analysis. Therefore, data analysts/scientists perform repetitive and time consuming tasks.

[0004] Software services in data science are time-consuming to configure. Time-savings gained by using software services to automate manual data processing tasks can be lost due to time overheads of configuring the software services to execute.

[0005] It is desirable to facilitate user access to data processing software services.

[0006] Embodiments include a data processing apparatus, the apparatus comprising: a software library, storing a plurality of software services, each executable software service being configured to execute a respective data processing function; a user interface configured to receive a plurality of user input commands, each user input command expressed in a domain specific language and defining a data processing target and a data processing request; and a parser. The parser is configured to extract from each user input command: the data processing request from the domain specific language; and the defined data processing target. The apparatus further comprises: a knowledge base, configured to maintain a record of the data processing request and the defined data processing target for each of the plurality of user input commands; a software service execution scheduler, configured, for each user input command, to obtain the data processing request from the parser, and to compile a schedule of one or more software services from among the plurality of software services to fulfil the data processing request; a software service execution controller configured, for each user input command, to control execution of the compiled schedule of one or more software services, the defined data processing target being the input data to the controlled execution, and to output a processing result of said controlled execution; and a result processor, configured to obtain the output processing result, and, based on the records of data processing requests and defined data processing targets maintained by the knowledge base, to identify a data processing request candidate for performance on the processing result, and to output to the user as a selectable user input command expressed in the domain specific language, via the user interface, the identified data processing request candidate with the processing result defined as a data processing target.

[0007] Advantageously, the data processing apparatus defined above enables a user to access the functionality of a plurality of software services by interacting with a single user interface in a domain specific language, obviating the need for the user to have knowledge of the syntax and semantics of the individual software services. Furthermore, the result processor recommends a next action to the user based on previous user actions, thus enabling the user to benefit from the experience of the apparatus. The result processor facilitates analysis of the processing target by proposing an option for additional processing of the processing result, based on previous data processing requests.

[0008] Data processing requests specify a desired data processing result, for example, by specifying a semantic descriptor of output data or by specifying a data type of output data. The software service execution scheduler converts the data processing request to a schedule of software services, which are in turn instructed to execute by the software service execution controller.

[0009] The domain specific language comprises controlled vocabulary that allows using natural language sentences describing operations in a very high level. It provides a natural way to interact with the data processing capabilities of the apparatus by using a language that is natural to the user.

[0010] The parser converts user inputs expressed in domain specific language to queries (data processing requests) for processing by the data processing apparatus.

[0011] The software services may be data science software services. That is, the data processing functions performed by the software services when executed are data science processing functions. Data science is the study & analysis of properties of the data itself.

[0012] The data processing apparatus may be a web server and the software services web services. The software services, whether web services or otherwise, may be microservices.

[0013] Each microservice implements (or wraps an external resource implementing) some atomic functionality in the system. Microservices may be annotated with meta-information about the semantics of the operations that they perform.

[0014] The parser may be configured to extract a data processing request from the domain specific language of the user input by at least: parsing the domain specific language into a series of domain specific language elements; querying a domain specific language map, said domain specific language map mapping each member of a vocabulary of domain specific language elements to a data processing request element, to obtain a data processing request element mapped to each member of the series of domain specific language elements; and combining the obtained data processing request elements to form the data processing request.

[0015] The domain specific language map translates from the domain specific language employed by the user

in interacting with the apparatus via the user interface, to a data processing request which can be used as the basis for a schedule of execution of software services by the software service execution scheduler.

[0016] Optionally, the software service execution scheduler is configured to maintain a software service registry, the software service registry comprising an entry for each of the plurality of software services, the entry identifying the respective software service and specifying a data processing function performed by the software service when executed; wherein the data processing functions are each specified as one or more data processing request elements to which the domain specific language elements are mapped; and wherein the software service execution scheduler is configured to select software services for inclusion in the schedule by matching data processing request elements from the data processing request to software services for which the respective data processing request element is included in the specified data processing function in the respective registry entry.

[0017] The data processing request elements are, for example, semantic descriptors of data processing functions. The software service registry entries may be generated by a system administrator of the apparatus upon addition of software services to the software library. Advantageously, using a common set of semantic descriptors for the data processing request elements to which user entries are mapped, and for the specification of data processing functions of software services, enables the software service execution controller to interpret the data processing request in terms of specific software services to execute.

[0018] In a particular implementation of the result processor, the result processor is further configured to output the obtained processing result to the user via the user interface.

[0019] The processing result may be output to the user in complete or summarised form. For example, the result processor may comprise processing logic for extracting a summary from a processing result. Software services may be configured to output processing results to a particular URL (uniform resource locator) or URI (uniform resource identifier) specified for the software service, so that the output processing result may be a link or reference to said URL or URI, or may comprise data copied therefrom.

[0020] Optionally, the records of data processing requests and defined data processing targets on which the identification of the data processing request candidate is based are constrained to records of data processing requests and defined data processing targets input to the user interface by the same user to which the identified data processing request candidate is to be output.

[0021] The data processing apparatus may further comprise a user authentication processor, configured to authenticate a user by username & password or by token.

[0022] Advantageously, the constraining to records of a particular user enable idiosyncratic user behaviour to be reflected in the proposed next action (the data processing request candidate) output to the user, and also protects a user from the idiosyncratic behaviour of others. Furthermore, it may be that users wish their actions to remain undiscoverable to others, and constraining to records of the same user prevents one user's actions being discoverable to another, albeit anonymously, in the form of a proposed next action.

[0023] Records are maintained by the knowledge base. In particular, the records of data processing requests and defined data processing targets maintained by the knowledge base may include, for each defined data processing target: one or more data types of data in the data processing target; the result processor being configured to identify a data processing request candidate for performance on the processing result by recognising a data type of data in the processing result, and identifying, as the candidate, a data processing request in a knowledge base record for a user input command in which the defined data processing target is recorded as including data of the recognised data type.

[0024] A data type being a particular kind of data item, as defined by the values it can take, the programming language used, or the operations that can be performed on it.

[0025] The result processor exploits knowledge of previous user actions to inform selections of actions (data processing request candidate) to propose to the user. The result processor therefore behaves as a machine learning mechanism, or artificial intelligence mechanism. Data type provides a characteristic by which to distinguish relevant from irrelevant records. For example, if a user typically initiates further analysis of time-series data, then this will be discovered by the result processor upon finding knowledge base records for the user in which the data processing target includes time-series data.

[0026] Furthermore, it may be that the identified data processing request candidate is selected by determining a most common data processing request among a relevant subset of the records maintained by the knowledge base, the relevant subset of records being those records for which a quantification of similarity between the characterisation of data in the defined data processing target and the characterisation of the data in the processing result is above a predefined threshold.

[0027] The characterisation may comprise values of one or more characteristics. The quantification of similarity may be, for example, a cosine distance of a vector representing the values. Non-numerical values may be mapped to numerical values for cosine distance comparison. As an alternative to cosine distance (or as an additional quantification to be combined with the cosine distance), the quantification of similarity may be the Mahalanobis distance. For example, the apparatus includes a support vector machine configured to accept two vectors representing values of characteristics as an input, and to output a value representing the Mahalanobis distance

between the two input vectors.

[0028] The user interface may be implemented by an interface including one or more from among the following for interacting with a user: a web interface; an application programming interface; a command line interface; a user voice command interface; and a graphical user interface.

[0029] Plural implementations of the user interface are listed above. Optionally, the system may be operable in a number of different modes, each mode being associated with a particular implementation of user interface.

[0030] The parser is configured to extract a data processing request from the user input. Such extracting may include, at least: extracting an incomplete data processing request from the user input; outputting to the user, via the user interface, a prompt for information to complete the incomplete data processing request; receiving, via the user interface, a response to the prompt from the user; and completing the incomplete data processing request with the received response.

[0031] Optionally, the software service execution scheduler is configured to: maintain a software service registry, the software service registry comprising an entry for each of the plurality of software services, the entry identifying the respective software service and specifying a data processing function performed by the software service when executed; divide the data processing request into a series of one or more instructed data processing functions; and compile an execution schedule, of one or more software services, from among the plurality of software services identified in the registry, to fulfil the respective data processing request by, for each of the one or more instructed data processing functions, identifying a software service for which the processing function specified in the registry matches the requested data processing function, and including the identified software service in the execution schedule.

[0032] The software service execution controller is configured to compile the schedule by at least: when more than one software services are identified for which the processing function specified in the registry matches one of the requested data processing functions, requesting a selection of one software service from among the more than one software services as manual selection candidates by a user of the apparatus, and receiving the requested selection from the user. Furthermore, the software service execution scheduler is configured to maintain a record of the compiling of the execution schedule for the respective instructed data processing function, including in the record the identity of the manual selection candidates and an indication of the received user selection; the software service execution scheduler being configured to automate the selection of one software service from among more than one software services identified for performing a requested data processing function based, at least partially, on the recorded indication of the received user selection from among manual selection candidates matching the more than one software services.

[0033] The software service execution scheduler makes a selection from among plural candidates based on maintained records of selections made by a user. Such records may be stored with a characterisation of the data processing target (i.e. values of one or more characteristics of the data: data size; data type; semantic representation of concept instantiated by the data) for processing by the software services from which the selection was made. The data processing target of the instructed data processing function is characterised in the same way (i.e. with values of the same characteristics), and hence the data characterisations provide a basis to discriminate between relevant and irrelevant records, with, for example, a most common selection among the relevant records being the automated selection. The discrimination between relevant and irrelevant may be achieved by imposing a threshold minimum cosine distance between vectors representing the characterisation of the data processing target of the instructed data processing function (for which an automated selection is sought) and the data processing target for which the record is maintained.

[0034] Advantageously, such a machine learning aspect to the software service execution scheduler enables intelligent scheduling of software services to satisfy a received data processing request in the absence of manual input. It is noted that manual inputs in satisfying previous data processing requests may be utilised by the machine learning mechanism.

[0035] Embodiments of another aspect include: a data processing method, comprising: storing a plurality of software services, each software service being configured to execute a respective data processing function; receiving, via a user interface, a plurality of user input commands, each user input command expressed in a domain specific language and defining a data processing target and a data processing request; extracting from each user input command: the data processing request from the domain specific language; and the defined data processing target. The method further comprises maintaining a record of the data processing request and the defined data processing target for each of the plurality of user input commands; and, for each user input command: obtaining the data processing request from the parser, compiling a schedule of one or more software services from among the plurality of software services to fulfil the data processing request, controlling execution of the compiled schedule of one or more software services, the defined data processing target being the input data to the controlled execution, and outputting a processing result of said controlled execution. The method further comprises obtaining the output processing result, and, based on the records of data processing requests and defined data processing targets maintained by the knowledge base, identifying a data processing request candidate for performance on the processing result, and outputting to the user via a user interface a selectable user input command expressed in the domain specific language, via the user

interface, the identified data processing request candidate with the processing result defined as a data processing target.

[0036] Embodiments of another aspect include a computer program which, when executed by a computing apparatus, causes the computing apparatus to function as a system defined in the claims as an invention embodiment.

[0037] Embodiments of another aspect include a computer program which, when executed by a computing apparatus, causes the computing apparatus to perform a method defined above or elsewhere in this document as an invention embodiment.

[0038] Furthermore, embodiments of the present invention include a computer program or suite of computer programs, which, when executed by a plurality of interconnected computing devices, cause the plurality of interconnected computing devices to operate as a system embodying the present invention.

[0039] In any of the above aspects, the various features may be implemented in hardware, or as software modules running on one or more processors. Features of one aspect may be applied to any of the other aspects.

[0040] The invention also provides a computer program or a computer program product for carrying out any of the methods described herein, and a computer readable medium having stored thereon a program for carrying out any of the methods described herein. A computer program embodying the invention may be stored on a computer-readable medium, or it could, for example, be in the form of a signal such as a downloadable data signal provided from an Internet website, or it could be in any other form.

[0041] A detailed description of embodiments will now be provided, with reference to the accompanying drawings, in which:

Figure 1 illustrates a data processing apparatus;
Figure 2 illustrates a data processing method;
Figure 3 illustrates user inputs and messages output to the user by the data processing apparatus; and
Figure 4 illustrates a hardware configuration of a data processing apparatus.

[0042] Figure 1 illustrates a data processing apparatus 10. The data processing apparatus 10 comprises a user interface 11, a parser 13, a software service execution scheduler 14, a software service execution controller 15, a result processor 16, and a knowledge base 17; which collection of components may be collectively referred to as a virtual assistant, since those components provide intelligent (based on experience) support to a user in accessing the functionality provided by software services in the software library 12.

[0043] Figure 2 illustrates a data processing method of an embodiment.

[0044] A step of storing a plurality of software services, each software service being configured, upon execution, to execute a respective data processing function, is represented by step S201 in Figure 2. The line intersecting step S201 indicating the storage persists during performance of steps S202 to S208. The storage of software services in step S201 may be performed by the software library 12 of Figure 1.

[0045] The software library 12 stores a plurality of executable software services. For example, the software services may be web services, and whether web service or otherwise, the software services may be microservices.

[0046] A microservice is an atomic service in a data processing apparatus. Atomic in this context means single responsibility or single function. A microservice is distinguished from a generic web service by the dimension of service. For example, a generic web service would include some form of authentication as part of a wider functionality. In a microservice-based apparatus, authentication is a dedicated microservice.

[0047] The software services, whether microservices or otherwise, may be RESTful software services, each defining methods for GET, and POST and/or PUT requests.

[0048] REST (Representational State Transfer) is an architectural style which governs the proper behaviour of participants in the web for machines. REST sets out constraints for system architectures to which conforming is described as being 'RESTful', the first of which is that the architecture has a client-server arrangement, with clients being separated from servers by a uniform interface. There are four guiding principles of the interface between client and server, and an interface developed in accordance with these principles can be described as 'RESTful'. For example, an API can be written in accordance with the REST guiding principles for interfaces to the software services, and would hence be described as a 'RESTful API'. Such a restful API for a software service may be stored in a registry entry for a software service stored by the software service execution scheduler 14, or stored in a location made accessible (for example, by a reference) by a reference in said registry. HTTP as a protocol can be used in a RESTful manner, and RESTful HTTP is suitable for the web for machines. RESTful interfaces (APIs) are popular for a number of key reasons: there is simplicity of the basic protocol built on a proven foundation with solid architectural principles, and the result is approachable and usable by web developers.

[0049] In brief, the REST architectural style describes six constraints (one of the six is optional) on a system architecture are as follows:

- the architecture should be client-server;
- the client and server are separated by a uniform interface;
- the architecture is stateless, meaning that no client context is stored on the server between requests from the client - each request contains all of the information necessary to service the request, with

state information held in the client itself;
- clients are able to cache responses;
- (optional) functionality may be extended by a server transferring logic to a client which the client can execute.

[0050] In the context of the software service execution system, the client is the entity making the data processing request, and the server is the web server or other computing device executing the software services. The data processing apparatus 10, either as part of the software library 12 or otherwise, may comprise an execution platform for the plurality of software services, for example, a processor and memory to store the data being processed and execute the processing logic of the software service. The plurality of software services stored by the software library 12 may be standalone software services for execution by the data processing apparatus 10, or may wrap externally held and executed software services.

[0051] The guiding principles for the uniform interface are briefly summarised below:

- individual resources in the domain can be identified in requests from the client (this would be via URIs (Universal Resource Identifiers) in a web-based system). The resources themselves are separate entities from the representations returned to the client;
- the representation of a resource held by a client is sufficient to give the client enough information to modify or delete the resource on the server (permissions allowing);
- each message between client and server contains enough information for the recipient to process the message;
- the representation of a resource provided from the server to the client should include hypertext links to related resources.

[0052] A positive aspect of the REST architectural style is that it links well with information models, an information model being a formalised description of items in a domain and relationships between those items. The operations allowed in a RESTful API are constrained (fixed), this avoids the unwanted side effects of poor programming behaviour which would ordinarily lead to problems in linking an interface with an information model.

[0053] In fact, a RESTful API for a particular domain may be defined purely in terms of the information model for the domain, and by how this model then appears inside different data formats, the data formats being wire level (low level or implementation level) manifestations of the information model. Unfortunately, APIs currently in use show disparities regarding their approach to information modelling, how this appears inside data formats, and how the semantics of HTTP are brought to use in the specific domain of the API(s) in question. This lack of consistency is problematic since potential benefits of a RESTful protocols are lost, for example, the potential for re-usable toolkits (eg standard code) and generic client agents (equivalent to a browser).

[0054] The virtual assistant components may be realised by instructions stored on a memory and executed by a processor.

[0055] A step of receiving, via a user interface, a plurality of user input commands, each user input command expressed in a domain specific language and defining a data processing target and a data processing request; is represented by S202 in Figure 2. The receiving user commands step S202 may be performed by the user interface 11 of Figure 1.

[0056] The user interface 11 is configured to receive a plurality of user input commands, each user input command at least partially expressed in a domain specific language and defining a data processing target and a data processing request. The data processing target may be defined by a URI or URL at which the data processing target (i.e. some data such as a file) is accessible. The domain specific language may define a data processing target by some syntax or semantic information indicating that a next part of the user input is a data processing target, for example, [VERB] [TARGET], so that the use of a verb in the domain specific language of the user input indicates that a data processing target is to follow.

[0057] Domain specific language is a vocabulary of terms which are recognisable among a natural language input, and which map to specific elements of data processing requests in the apparatus 10.

[0058] The user interface 11 interacts with the user to enable user input commands to be input to the apparatus 10. The user interface 11 comprises at least one means for interacting with a user. For example, the user interface includes at least one of: a web interface; an application programming interface; a command line interface; a user voice command interface; and a graphical user interface. The user input received by the user interface 11 is at least partially expressed in domain specific language. Complete absence of domain specific language from user input received by the user interface 11 may cause the user interface to output a message requesting a new input. It is noted that the parser 13 may be required to notify the user interface 11 of the absence or otherwise of domain specific language from the user input.

[0059] The user interface 11 also provides a means to output processing results, in full or summarised form, to the user. The user interface 11 also provides a means to propose a next data processing action to the user, in the form of a data processing request candidate, expressed in domain specific language (and therefore understandable to the user) and selectable by the user as a user input. For example, such a data processing request candidate may be output in a sentence such as: "would you like me to [semantic representation of data processing function] the [definition of data processing target]?", to which a "yes" from the user would be sufficient to initiate execution of the software service performing the data processing function on the defined data process-

ing target.

[0060] A step of extracting from each user input command: the data processing request from the domain specific language; and the defined data processing target; is represented by step S203 in Figure 2. It is noted that the term parameters is used as shorthand for the extracted data processing request & data processing target. The extracting parameters from user commands step S203 of Figure 2 may be performed by the parser 13 of Figure 1.

[0061] The parser 13 is configured to extract information from the user input that can be used to control execution of software services. Noting that the software services each perform a single data processing function on input data, the information required to extract is a requested data processing function (or functions) and a location from which the input data for processing is accessible. In particular, the parser 13 is configured to extract from each user input command: the data processing request from the domain specific language; and the defined data processing target.

[0062] The parser 13 serves as a translator between the domain specific language in which user inputs are expressed, and the vocabulary of the software service execution scheduler, which selects software services to execute in order to fulfil the data processing request defined by the user. To that end, the parser 13 may store a domain specific language map, said domain specific language map mapping each member of a vocabulary of domain specific language elements to a data processing request element.

[0063] The parsed data processing request and data processing target are output by the parser 13 and obtained by both the software service execution scheduler 14 and knowledge base 17.

[0064] A step of maintaining a record of the data processing request and the defined data processing target for each of the plurality of user input commands; is represented by step S204 of Figure 2. Of course, the maintaining is persistent throughout multiple executions of the method. The maintaining a record of user commands and parameters step S204 of Figure 2 may be performed by the knowledge base 17 of Figure 1.

[0065] The knowledge base 17 therefore receives a record, for storage, of data processing instructed by a user, and the data upon which said data processing is instructed. The data may be characterised in that record by, for example, one or more data types of data items among the data.

[0066] A step of obtaining the data processing request from the parser, and compiling a schedule of one or more software services from among the plurality of software services to fulfil the data processing request; is represented by step S205 in Figure 2. The compiling a schedule step S205 may be performed by the software service execution scheduler 14 of Figure 1.

[0067] The parsed data processing request defines one or more data processing functions. It may be that some processing of the parsed data processing request is performed by the software service execution scheduler 14 in order to determine the constituent data processing function(s). That is to say, the constituent data processing function(s) may be implicitly defined by the parsed data processing request and extracted (i.e. made explicit) by the software service execution scheduler 14, for example, by reference to the software services themselves or to a registry of software services maintained by the service execution scheduler 14. For example, the parsed data processing request may be "summarize" and the data processing target may be plural disparate relational databases. A software service has a registry entry indicating that it performs a summarize data processing function. However, the registry entry specifies that input data is to be stored locally as a single table. Another software service has a registry entry indicating that it performs a load data processing function, with input data being an external database and a data processing result being a locally stored table of data. Another software service has a registry entry indicating that it performs a table join on locally stored tables, taking plural locally stored tables as inputs and generating as a processing result a single locally stored table. This complex example exemplifies a technique for compiling a software service execution schedule.

[0068] In summary, the data processing request may be defined in terms of a processing result. The software service execution scheduler 14 stores a registry of software services of the software library 12, defined in terms of the input data on which they are operable, and the output generated (for example, both input and output defined by a data type or semantic descriptor thereof). The software service execution controller 14, upon identifying a type of data of the parsed data processing target, then compiles a schedule by selecting software services in a stepwise fashion to go from the type of data of the parsed data processing target to the type of data of the processing result of the data processing request.

[0069] In a more simple example, each parsed data processing request may correspond to a single data processing function performed by a particular software service, and hence compiling the schedule is simply a process of matching the data processing function to the software service.

[0070] An order in which the constituent data processing functions are to be performed in order to fulfil the data processing request may also be defined by the parsed data processing request.

[0071] A schedule, which may also be referred to as an execution schedule or execution plan, is a plan identifying which software services to execute, in which order (noting that the order may include parallel execution of plural software services), and with which input data. The compiling of the schedule and control of execution are not necessarily steps performed serially. It may be that the compiling is adaptive rather than prescribed or predetermined. The compilation of an element of the execution schedule may be dependent upon the outcome of

execution of preceding software services in the execution schedule. The schedule is output by the software service execution scheduler 14 and obtained by the software service execution controller 15. It is noted that processing results from the software services may be input to the software service execution scheduler 14 for use in adaptive compilation.

[0072] The schedule is compiled by the software service execution scheduler 14 in order to fulfil the parsed data processing request, for example, by using annotations about the data processing functions of the software services, which annotations may be stored in a registry. For example, it may be that the parsed data processing request specifies a particular data type sought as a processing result, for example, specifying the particular data type semantically or by use of a particular filename or group of filenames. The execution schedule is compiled by the software service execution scheduler 14 by determining a software service or series of software services stored by the software library 12 which, when executed (in the case of a series, with proceeding software services in the series taking as an input the processing result(s) of one or more preceding software service(s) in the series) transform the data processing target output by the parser 13 into the particular data type sought as a processing result and specified by the parsed data processing request.

[0073] Each of the software services is configured to execute a respective data processing function. The data processing executed by each of the software services transforms input data of a particular type into output data of a particular type. The exception to this singular input type to output type relationship is the event of a failure of the software service, such as a timeout, which does not generate output data of the output type. Type means a type of data, and may be specified by, for example, a single or group of filename extensions or file types, and/or by a semantic descriptor or descriptor(s) of the data.

[0074] A step of controlling execution of the compiled schedule of one or more software services, the defined data processing target being the input data to the controlled execution; is represented by step S206 in Figure 2. The controlling execution of the schedule step S206 may be performed by the software service execution controller 15 of Figure 1. A step of outputting a processing result of the controlled execution is represented by S207 in Figure 2, and may be performed by the software service execution controller 15 of Figure 1.

[0075] The schedule is output by the software service execution scheduler 14 and obtained by the software service execution controller 15. The software service execution controller 15 is configured, for each user input command, to control execution of the compiled schedule of one or more software services, the defined data processing target being the input data to the controlled execution, and to output a processing result of said controlled execution. Controlling execution of the schedule by the software service execution controller 15 is a process of issuing calls to the respective software services to execute, in the order/timing determined by the schedule, and with input data specified by the schedule. The line marked "execution control" between the software service execution controller 15 and the software library 12 in Figure 1 illustrates the control of execution. For example, controlling execution comprises calling the or each of the software services included in the schedule to perform the respective data processing function on input data specified in the call. The input data to the first of a series of software services in the schedule is the data processing target defined in the user input, with the input data of subsequent software services in the schedule being the processing result of the adjacent preceding software service in the schedule. Input data may be specified by reference to a URI or URL from which it is accessible.

[0076] Figure 2 illustrates that the processing result may be output from the apparatus (i.e. to the user) and also to a further process within the apparatus.

[0077] An outcome of the execution may be returned to the software service execution scheduler 14 in response to the execution call, for use in compiling the remainder of the schedule for the respective user input. The outcome may be, for example, an indication of whether a processing result was generated or whether the execution timed out. Optionally, the response may include information such as a measurement of the size of data output by the executed software service as a processing result.

[0078] Optionally, the software service execution scheduler 14 is configured to revert to the user, via the user interface 11, to select between plural software services suitable for performing a particular data processing function.

[0079] The software service execution scheduler 14 may store a registry of software services with an entry per software service, the entry identifying the software service and defining the data processing function performed by the software service when executed. Said definition may be in terms of a semantic descriptor of the data processing function, which may be in domain specific language, or in language which is mapped to domain specific language. Said definition may be in terms of type of input data and type of output data. A match between a data processing function specified for a software service in the registry and a data processing function forming part of a data processing request (an instructed data processing function) is an indication that the software service for which the processing function is specified is suitable to perform the instructed data processing function. The match is based on the definition of the instructed data processing function, and the data processing function specified for the software service in the registry. The match may be based on semantics, for example, the data processing function is defined semantically in the registry (for example, "tokenize"; "load"; "summarize"; "compare") and the instructed data processing function is de-

fined semantically using the same or similar term(s), so that a semantic comparison between the semantic definition of the requested data processing function and the sematic definition of the data processing function yielding a similarity score above a threshold is taken to be a "match". Alternatively or additionally, the definition of the instructed data processing function may be defined in terms of input data and requested processing result, for example, defining each by a semantic descriptor of the data type ("tabular data" "matrix" "document" "vector") or by a filename extension. The data processing functions specified in the registry may be defined in the same way, so that a match is matching input and output filename extensions, or matching (i.e. semantic similarity above a threshold) input and output semantic data type descriptors.

[0080] The semantic data type descriptor is a semantic representation of the concept instantiated by the data.

[0081] A step of obtaining the output processing result, and, based on the records of data processing requests and defined data processing targets maintained by the knowledge base, identifying a data processing request candidate for performance on the processing result, and outputting to the user via a user interface a selectable user input command expressed in the domain specific language, via the user interface, the identified data processing request candidate with the processing result defined as a data processing target; is represented by step S208 of Figure 2. The identifying and outputting request candidate step S208 of Figure 2 may be performed by the results processor 17 of Figure 1.

[0082] Once execution of the schedule is complete, the processing result is output to the result processor 16. For example, in the case of the schedule defining a series of software services in which each software services takes as input data the processing result of the preceding in the series, the processing result may be the processing result of the final software service in the series. Although not specifically illustrated in Figure 1, the result processor 16 may be notified of the parsed data processing request, and in particular, any particular element of the parsed data processing request indicating a form or destination of output data sought in response to the user input.

[0083] The result processor 16 is configured to obtain the output processing result, and, based on the records of data processing requests and defined data processing targets maintained by the knowledge base, to identify a data processing request candidate for performance on the processing result, and to output to the user as a selectable user input command expressed in the domain specific language, via the user interface, the identified data processing request candidate with the processing result defined as a data processing target. In other words, the result processor 16 refers to the records of which data processing requests have been carried out on which target data, to determine a next data processing request to propose to the user.

[0084] In a simple example, the record of the data processing target in the knowledge base includes a value of one or more characterisations of the data processing target. Such characterisations may include: amount of data; data type (in terms of syntax); semantic descriptor (semantic representation of concept instantiated by the data). The result processor 16 obtains values of one or more of said characterisations for the processing result. It is noted that a single processing result may contain different types of data and data having different semantic descriptors. In which case, multiple characterisations each apply to respective parts of the processing result (for example, each column), and the procedure of finding a candidate data processing request can be performed multiple times for the single processing result.

[0085] The characterisation of (all or part of) the processing result can be used in many ways to inform the selection of a data processing request candidate. Two particular techniques will be set out.

[0086] In a first technique, the processing result characterisation is compared with recorded processing target characterisations (for example, using a vector distance comparison) to obtain a quantification of the similarity between the processing result characterisation and each of the processing target characterisations. A threshold is set (which may be predefined or set at a fixed proportion) and only those recorded processing target characterisations for which the quantification of similarity exceeds the threshold are considered. Of those considered, the most common corresponding data processing request (noting that each record includes a data processing request and a data processing target) is determined and selected for proposal to the user as the data processing request candidate.

[0087] In a second technique, the processing result characterisation is compared with recorded processing target characterisations (for example, using a vector distance comparison) to obtain a quantification of the similarity between the processing result characterisation and each of the processing target characterisations. A contribution to a frequency count to find the most common corresponding data processing request is weighted according to the quantification of similarity. So that, for example, a data processing request commonly performed on data characterised very differently to the processing result builds up a relatively small frequency count, whereas a data processing request carried out less commonly, but on data characterised very similarly to the processing result, builds up a relatively larger frequency count (noting that the processing result is the data upon which the next data processing request is to be performed).

[0088] The data processing request candidate selected by the result processor 16 using the records maintained by the knowledge base 17, is output to the user via the user interface, expressed in domain specific language, and identifying (all or part of) the processing result as the processing target. The user is presented with a proposal for a next data processing request, arrived at based on experience of previous user actions, which is

selectable by the user for parsing, scheduling, and execution in the same way as other user inputs.

[0089] Figure 3 illustrates exemplary communications between the user and the data processing apparatus. The data processing apparatus is represented by the initials AIDA: Artificial Intelligence for Data Analytics. The data processing functions of the software services in the example of Figure 3 are data analytics processing functions.

[0090] The user interface in the example of Figure 3 is an interactive shell. Other forms of communication could also be employed, for example, e-mails, a chat window, an interactive webpage, voice commands (as usual for many assistants), gestures, etc.

[0091] The example of Figure 3 exemplifies various elements of the functionality of the data processing apparatus 10.

[0092] In line 1 the user initiates a session. The term "Hi" is an example of domain specific language and is interpreted by the parser 13 as initiating a user session.

[0093] In line 2, the result of user authentication is output by the user interface 11, with the user ID "Dave" retrieved via an authentication process. In the present example, authentication processing is performed by the data processing apparatus 10 in a manner that is transparent to the user, and is token-based. The user-authentication is triggered by the user input of line 1, initiating a user session. If the user cannot be authenticated, then it is possible that the user is a new user. For a new user, a combination of user and password is requested.

[0094] The data processing apparatus 10 stores user preferences on a per user basis, which are loaded and implemented upon authentication. Following authentication, machine learning algorithms, which use records maintained by the knowledge base to select data processing request candidates for presentation to the user, may be trained or otherwise prepared for execution using records relating to past data processing requests from the authenticated user. For example, such records may be loaded into a cache accessible by the machine learning algorithm.

[0095] The software services perform data processing functions for data science and data analysis. Information output by the data processing apparatus 10 in response to questions posed by the user, such as in line 3, is tailored to provide information useful to a data science professional. For example in line 4 the user interface outputs to the user accurate information that could be used for support purposes.

[0096] The user inputs are parsed by the parser 13. In the example of line 3, the term "tell me" is not recognised as domain specific language and is ignored by the parser 13. The term "what is your name", on the other hand, is domain specific language which maps to an instruction to output system information about the data processing apparatus 10 and specifically the virtual assistant components thereof.

[0097] When information is omitted by the user, so that a user input defines only a partial data processing request, or does not unambiguously specify a data processing target, the parser 13 performs processing to:

Infer the missing information when possible.

Display via the user interface a set of possibilities in ambiguous contexts.

Output prompts to the user for the missing information via the user interface when previous options are not feasible.

Output via the user interface a request to the user to reformulate the sentence when all of the above fails.

[0098] In the example of line 15, the parser 13 identifies "it" as the data processing target, by processing the input sentence, using natural language processing and the domain specific language mapping. Via inference, using an inference rule which specifies if data processing target not specified, infer that the most recently input data processing target specified by the user is the specified data processing target. The parser 13 may revert to the user for confirmation of inferred information via the user interface 11.

[0099] Line 5 illustrates the functionality of the virtual assistant components to function as a very high level programming language. Via the user interface 11 and the parser 13, user inputs are received which define variables and assign a value to them. In the example of line 5, the data processing target is specified as http://example.com/file.csv, and the "as" is recognised as domain specific language which the parser 13 is aware is followed by a name to be assigned to the data stored at the specified URL. The parser 13, when parsing future user inputs in the same session, interprets "dataset1" as defining the data held at the specified URL.

[0100] The term "explore" utilised by the user causes a number of software services to be scheduled by the software service execution scheduler 14. The term "explore" is domain specific language, which is interpreted by the parser 13 as a data processing request to summarize the data processing target. Summarize is a data processing function of one or more of the software services, which is known by the software service execution scheduler 14 to correspond to a data processing request to generate a summary. The software service execution scheduler 14 is configured to compile an execution schedule to output a summary of the data processing target. Based on information held by the software service execution scheduler 14, there is no software service which can generate a processing result of a summary with an input of a remotely held csv file. However, there is a software service for which the registry entry indicates a processing result of a summary from an input of a locally held csv file, and another software service for which the registry entry indicates a processing result of a locally held csv file from an input of a remotely held csv file.

[0101] The compiled schedule is output to the software service execution controller 15, and the software services

executed on the data processing target. The processing result is output to the user via the result processor 16 in line 6. The summary displayed in line 6 is representative, noting that software services may generate much more detailed summaries of data processing targets.

[0102] In a more complex example, an instruction to explore a URL that points to a webpage with unknown content may be handled differently by the software service execution scheduler 14, operating adaptively. First scheduling execution of a software service to establish the protocol or protocols for accessing the content of the webpage, and then, following execution of said software service, receiving a processing result from the software service (defining one or more protocols), and then compiling a new schedule of software services based on the defined protocol or protocols. In this manner, the data processing apparatus 10 uses software services stored in the software library 12 to handle several different protocols in a way that is transparent to the user (http, https, ftp, sftp, etc.). The data processing apparatus 10 provides the user with an intuitive interface based on natural language structure and domain specific language elements.

[0103] Line 6 demonstrates that the result processor 16 characterises the processing result (on a per row or per column basis), and uses the characterisation as a key to find relevant previously recorded (in the knowledge base 17) data processing requests from the same user, finding records of data processing targets sharing a characterisation with the characterisation of the present processing result, and the data processing request of the user in the recorded instances. A machine learning algorithm searches for and finds relevant records using data characterisation as a key, and determines, from among the data processing requests in the relevant records, which to output to the user as a data processing request candidate (exemplified in lines 6, 8, 10). The basis on which the machine learning mechanism determines one data processing request from plural, if plural are defined in relevant records, is implementation dependent. In a first exemplary technique, the machine learning algorithm may simply find the most recent record (assuming the records are time-stamped), or the most popular among the most recent n (for example, n=10) records. In a second exemplary technique, the machine learning algorithm may quantify the relevance of each record based on similarity of characterisation of the recorded data processing target to the processing result in question (wherein a threshold minimum of the quantification is imposed for a record to be deemed relevant), and a score calculated for each recorded data processing request, wherein the score is a count of relevant records in which the data processing request is defined. Optionally, the contribution to the count made by a record may be weighted according to the respective quantification of relevance.

[0104] By characterising the data of all or part of the processing result, the results processor 16 can be seen as understanding the contents of the file, and using a machine learning algorithm intelligently uses the previous decisions of the user to suggest a next data processing operation based on what makes sense in the current context (i.e. context provided by data characterisation and the previous decisions and preferences of the user). In this particular example the system detects the type of one column in the dataset and runs a time series analysis.

[0105] Line 10 demonstrates that the result processor 16 may also output a representation of the rationale for outputting a particular data processing request candidate. In addition, line 10 exemplifies that the data processing functions performed on datasets have a very high level, as for instance training a regression model.

[0106] Line 13 illustrates accessibility of the persistence layer to the data processing apparatus 10. A software service from the software library 12 is configured to perform an operation of writing a data processing target to the persistence layer upon a "store" user input. The persistence layer may be accessed upon the express instruction of the user as in the example of line 13, or may be automatically used by other software services in the absence of an express instruction from the user, depending on the operation performed, the characteristics of the data and the previous preferences of the user. If the information that the system has is not enough to make a decision in a given context, it will prompt a question to the user via the user interface 11.

[0107] Line 15 demonstrates syntax for separating sequences of operations using commas, consistent with natural language syntax. Without prior configuration, the system provides freedom to use other separators and combinations of them, as for example: ',', '.', 'and', or 'then'.

[0108] As another example of the intelligence of the data processing apparatus 10, in line 16 it can be seen that when the execution of a software service under the control of the software service execution controller 16 software takes, or is expected to take, over a predefined threshold amount of time, the task is automatically sent to the background, the user is notified via the user interface 11 (see line 16), and new user inputs can be submitted during execution.

[0109] There may be cases in which no relevant records can be found by the result processor 16, that is to say, based on characterisation of all or a part of the processing result, there are no records in the knowledge base 17 of data processing requests having similarly characterised data as a data processing target. In line 8 we see that the user is asked for confirmation on an operation. For example, in the absence of relevant records, such a proposal may instead be based on registry entries for software services defining input data characterised in the same or similar way as the processing result in question. After several iterations, the results processor may use the machine learning capabilities to proactively perform the operations that are usually done with similarly characterised data, or skip the questions for the opera-

tions that are never used.

[0110] FIGURE 4 is a block diagram of a computing device, such as a web server, which embodies the present invention, and which may be used to implement a data processing method of an embodiment. The computing device comprises a processor 993, and memory, 994. Optionally, the computing device also includes a network interface 997 for communication with other computing devices, for example with other computing devices of invention embodiments.

[0111] For example, an embodiment may be composed of a network of such computing devices. Optionally, the computing device also includes one or more input mechanisms such as keyboard and mouse 996, and a display unit such as one or more monitors 995. The components are connectable to one another via a bus 992.

[0112] The memory 994 may include a computer readable medium, which term may refer to a single medium or multiple media (e.g., a centralized or distributed database and/or associated caches and servers) configured to carry computer-executable instructions or have data structures stored thereon. Computer-executable instructions may include, for example, instructions and data accessible by and causing a general purpose computer, special purpose computer, or special purpose processing device (e.g., one or more processors) to perform one or more functions or operations. Thus, the term "computer-readable storage medium" may also include any medium that is capable of storing, encoding or carrying a set of instructions for execution by the machine and that cause the machine to perform any one or more of the methods of the present disclosure. The term "computer-readable storage medium" may accordingly be taken to include, but not be limited to, solid-state memories, optical media and magnetic media. By way of example, and not limitation, such computer-readable media may include non-transitory computer-readable storage media, including Random Access Memory (RAM), Read-Only Memory (ROM), Electrically Erasable Programmable Read-Only Memory (EEPROM), Compact Disc Read-Only Memory (CD-ROM) or other optical disk storage, magnetic disk storage or other magnetic storage devices, flash memory devices (e.g., solid state memory devices).

[0113] The processor 993 is configured to control the computing device and execute processing operations, for example executing code stored in the memory to implement the various different functions of the software library, user interface, parser 13, knowledge bade, software service execution scheduler, software services execution controller, and result processor, described here and in the claims. The memory 994 stores data being read and written by the processor 993. As referred to herein, a processor may include one or more general-purpose processing devices such as a microprocessor, central processing unit, or the like. The processor may include a complex instruction set computing (CISC) microprocessor, reduced instruction set computing (RISC) microprocessor, very long instruction word (VLIW) microprocessor, or a processor implementing other instruction sets or processors implementing a combination of instruction sets. The processor may also include one or more special-purpose processing devices such as an application specific integrated circuit (ASIC), a field programmable gate array (FPGA), a digital signal processor (DSP), network processor, or the like. In one or more embodiments, a processor is configured to execute instructions for performing the operations and steps discussed herein.

[0114] The display unit 997 may display a representation of data stored by the computing device and may also display a cursor and dialog boxes and screens enabling interaction between a user and the programs and data stored on the computing device. The input mechanisms 996 may enable a user to input data and instructions to the computing device.

[0115] The network interface (network I/F) 997 may be connected to a network, such as the Internet, and is connectable to other such computing devices via the network. The network I/F 997 may control data input/output from/to other apparatus via the network. Other peripheral devices such as microphone, speakers, printer, power supply unit, fan, case, scanner, trackerball etc may be included in the computing device.

[0116] The software library 12 of Figure 1, and the storing a plurality of software services step S201 of Figure 2, may be a processor 993 (or plurality thereof) executing processing instructions (a program) stored on a memory 994 and exchanging data via a network I/F 997. In particular, the processor 993 executes processing instructions to receive, via the network I/F or otherwise, execution controls from the software service execution controller 15 and execute the software services as indicated by the received execution controls, as indicated by the "execution controls" arrow in Figure 1. Furthermore, the processor 993 may execute processing instructions to store software services in the software library 12 on a connected storage unit. Furthermore, the processor 993 may execute processing instructions to transmit, via the network I/F 997, processing results to the result processor 16 for processing.

[0117] The user interface 11 of Figure 1, and the receiving user commands step S202 of Figure 2, may be a processor 993 (or plurality thereof) executing processing instructions (a program) stored on a memory 994 and exchanging data with a user via a network I/F 997 or other input means. In particular, the processor 993 executes processing instructions to receive, via the network I/F or other input means, user input from the user and transmit the user input to the parser 13, as indicated by the "user input" arrow in Figure 1. In addition, the user interface executes processing instructions to output the data processing request candidate to the user as a selectable user input command.

[0118] The parser 13 of Figure 1, and the extracting parameters from user commands step S203 of Figure 2, may be a processor 993 (or plurality thereof) executing

processing instructions (a program) stored on a memory 994 and exchanging data with the user interface 11 and software service execution scheduler 14 via a network I/F 997 or another means of data exchange. In particular, the processor 993 executes processing instructions to receive from the user interface 11, the user input command, to extract the data processing request and the defined data processing target from the user input command, and output the extracted information to the knowledge base 17 and software service execution scheduler, as indicated by the arrows marked "data processing request" and "data processing target" in Figure 1. Furthermore, the processor 993 may execute processing instructions to store the extracted information on the knowledge base as a record.

[0119] The knowledge base 17 of Figure 1, and the maintaining a record of user commands and parameters step S204 of Figure 2, may be a processor 993 (or plurality thereof) executing processing instructions (a program) stored on a memory 994 and exchanging data via a network I/F 997 or another means of data exchange. In particular, the processor 993 executes processing instructions to receive, via the network I/F, information extracted from user inputs by the parser 13 from the parser 13, and to store the extracted information as a record. Furthermore, the records maintained by the knowledge base 17 are accessible by the result processor 16 in identifying a data processing request candidate. Furthermore, the processor 993 may execute processing instructions to store maintained records on a connected storage unit and/or to transmit, via the network I/F 997, maintained records relevant to a particular user (and processing result) to the result processor 16 for processing.

[0120] The software service execution scheduler 14 of Figure 1, and the compiling a schedule step S205 of Figure 2, may be a processor 993 (or plurality thereof) executing processing instructions (a program) stored on a memory 994 and exchanging data via a network I/F 997 or another means of data exchange. In particular, the processor 993 executes processing instructions to receive, via the network I/F or otherwise, a data processing request and data processing target extracted from a user input by the parser 13, and compile a schedule of software services to fulfil the data processing request with the data processing target as input data, and to output the schedule, as indicated by the "schedule" arrow in Figure 1. Furthermore, the processor 993 may execute processing instructions to store the schedule on a connected storage unit and/or to transmit, via the network I/F 997, the schedule to the software service execution controller 15 for execution.

[0121] The software service execution controller 15 of Figure 1, the controlling execution step S206 of Figure 2, and the outputting step S207 of Figure 2, may be a processor 993 (or plurality thereof) executing processing instructions (a program) stored on a memory 994 and exchanging data via a network I/F 997 or another form of data exchange. In particular, the processor 993 executes processing instructions to receive, via the network I/F or another form of data exchange, the schedule output by the software service execution schedule, and to control execution of the schedule by the software services of the software library, as indicated by the "execution control" arrow of Figure 1.

[0122] The result processor 16 of Figure 1, and the identifying and outputting a request candidate S207 of Figure 2, may be a processor 993 (or plurality thereof) executing processing instructions (a program) stored on a memory 994 and exchanging data via a network I/F 997 or another form of data exchange. In particular, the processor 993 executes processing instructions to receive, via the network I/F or otherwise, processing results from the software services of the software library 12 executed under the control of the software service execution controller 15 and identify a data processing request candidate from among records held by the knowledge base (using a machine learning algorithm). Furthermore, the processor 993 may execute processing instructions to store the identified data processing request candidate on a connected storage unit and/or to transmit, via the network I/F 997 or otherwise, the identified data processing request candidate to the user via the user interface 11.

[0123] Methods embodying the present invention may be carried out on a computing device such as that illustrated in Figure 4. Such a computing device need not have every component illustrated in Figure 4, and may be composed of a subset of those components. A method embodying the present invention may be carried out by a single computing device in communication with one or more data storage servers via a network. The computing device may be a data storage itself storing the knowledge base 17 and processing results.

[0124] A method embodying the present invention may be carried out by a plurality of computing devices operating in cooperation with one another. One or more of the plurality of computing devices may be a data storage server storing at least a portion of the knowledge base 17 and processing results.

Claims

1. A data processing apparatus, comprising:

a software library, storing a plurality of software services, each software service being configured to execute a respective data processing function;

a user interface configured to receive a plurality of user input commands, each user input command expressed in a domain specific language and defining a data processing target and a data processing request;

a parser configured to extract from each user input command:

the data processing request from the domain specific language; and
the defined data processing target;

a knowledge base, configured to maintain a record of the data processing request and the defined data processing target for each of the plurality of user input commands;
a software service execution scheduler, configured, for each user input command, to obtain the data processing request from the parser, and to compile a schedule of one or more software services from among the plurality of software services to fulfil the data processing request;
a software service execution controller configured, for each user input command, to control execution of the compiled schedule of one or more software services, the defined data processing target being the input data to the controlled execution, and to output a processing result of said controlled execution; and
a result processor, configured to obtain the output processing result, and, based on the records of data processing requests and defined data processing targets maintained by the knowledge base, to identify a data processing request candidate for performance on the processing result, and to output to the user as a selectable user input command expressed in the domain specific language, via the user interface, the identified data processing request candidate with the processing result defined as a data processing target.

2. A data processing apparatus according to claim 1, wherein:

the parser is configured to extract the data processing request from the domain specific language of the user input by at least:

parsing the domain specific language into a series of domain specific language elements;
querying a domain specific language map, said domain specific language map mapping each member of a vocabulary of domain specific language elements to a data processing request element, to obtain a data processing request element mapped to each member of the series of domain specific language elements;
combining the obtained data processing request elements to form the data processing request.

3. A data processing apparatus according to claim 2, wherein

the software service execution scheduler is configured to maintain a software service registry, the software service registry comprising an entry for each of the plurality of software services, the entry identifying the respective software service and specifying a data processing function performed by the software service when executed; wherein
the data processing functions are each specified as one or more data processing request elements to which the domain specific language elements are mapped; and
the software service execution scheduler is configured to select software services for inclusion in the schedule by matching data processing request elements from the data processing request to software services for which the respective data processing request element is included in the specified data processing function in the respective registry entry.

4. A data processing apparatus according to any of the preceding claims, the result processor being further configured to output the obtained processing result to the user via the user interface.

5. A data processing apparatus according to any of the preceding claims, wherein
the records of data processing requests and defined data processing targets on which the identification of the data processing request candidate is based are constrained to records of data processing requests and defined data processing targets input to the user interface by the same user to which the identified data processing request candidate is to be output.

6. A data processing apparatus according to any of the preceding claims, wherein
the records of data processing requests and defined data processing targets maintained by the knowledge base include, for each defined data processing target:

a characterisation of data in the data processing target;
the result processor being configured to identify a data processing request candidate for performance on the processing result by characterising some or all data in the processing result, and identifying, as the candidate, a data processing request in a knowledge base record for a user input command in which the characterisation of data in the defined data processing target is the same or similar to the characterisation of the data in the processing result.

7. A data processing apparatus according to claim 6, wherein
the identified data processing request candidate is

selected by determining a most common data processing request among a relevant subset of the records maintained by the knowledge base, the relevant subset of records being those records for which a quantification of similarity between the characterisation of data in the defined data processing target and the characterisation of the data in the processing result is above a predefined threshold.

8. A data processing apparatus according to any of the preceding claims, wherein the user interface comprises one or more from among:

a web interface;
an application programming interface;
a command line interface;
a user voice command interface; and
a graphical user interface.

9. A data processing apparatus according to any of the preceding claims, wherein the parser is configured to extract a data processing request from the user input by at least:

extracting an incomplete data processing request from the user input;
outputting to the user, via the user interface, a prompt for information to complete the incomplete data processing request;
receiving, via the user interface, a response to the prompt from the user; and
completing the incomplete data processing request with the received response.

10. A data processing apparatus according to any of the preceding claims, wherein the software service execution scheduler is configured to:

maintain a software service registry, the software service registry comprising an entry for each of the plurality of software services, the entry identifying the respective software service and specifying a data processing function performed by the software service when executed;
divide the data processing request into a series of one or more instructed data processing functions; and
compile an execution schedule, of one or more software services, from among the plurality of software services identified in the registry, to fulfil the respective data processing request by, for each of the one or more instructed data processing functions, identifying a software service for which the processing function specified in the registry matches the requested data processing function, and including the identified software service in the execution schedule.

11. A data processing apparatus according to claim 10, wherein
the compiling includes if more than one software services are identified for which the processing function specified in the registry matches one of the requested data processing functions, requesting a selection of one software service from among the more than one software services as manual selection candidates by a user of the apparatus, and receiving the requested selection from the user; and
the software service execution scheduler is configured to maintain a record of the compiling of the execution schedule for the respective instructed data processing function, including in the record the identity of the manual selection candidates and an indication of the received user selection;
the software service execution scheduler being configured to automate the selection of one software service from among more than one software services identified for performing a requested data processing function based, at least partially, on the recorded indication of the received user selection from among manual selection candidates matching the more than one software services.

12. A data processing method, comprising:

storing a plurality of software services, each software service being configured to execute a respective data processing function;
receiving, via a user interface, a plurality of user input commands, each user input command expressed in a domain specific language and defining a data processing target and a data processing request;
extracting from each user input command:

the data processing request from the domain specific language; and
the defined data processing target;

maintaining a record of the data processing request and the defined data processing target for each of the plurality of user input commands;
for each user input command:

obtaining the data processing request from the parser,
compiling a schedule of one or more software services from among the plurality of software services to fulfil the data processing request,
controlling execution of the compiled schedule of one or more software services, the defined data processing target being the input data to the controlled execution, and
outputting a processing result of said controlled execution; and

obtaining the output processing result, and, based on the records of data processing requests and defined data processing targets maintained by the knowledge base, identifying a data processing request candidate for performance on the processing result, and outputting to the user via a user interface a selectable user input command expressed in the domain specific language, via the user interface, the identified data processing request candidate with the processing result defined as a data processing target.

13. A computer program which, when executed by a computing apparatus, causes the computing apparatus to perform a method according to claim 12.

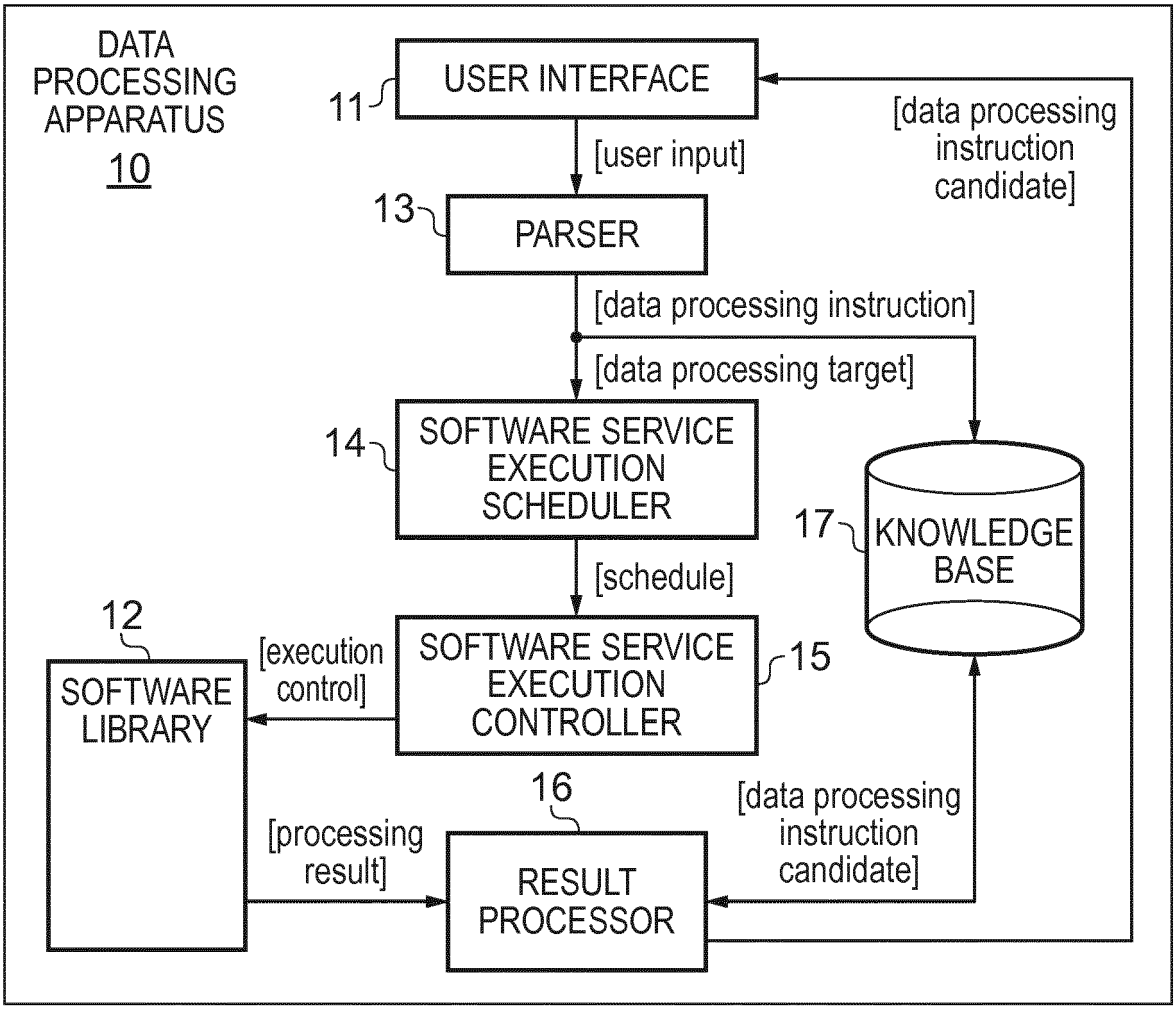
DATA PROCESSING APPARATUS
10
11
USER INTERFACE
[user input]
[data processing instruction candidate]
13
PARSER
[data processing instruction]
[data processing target]
14
SOFTWARE SERVICE EXECUTION SCHEDULER
17
KNOWLEDGE BASE
[schedule]
12
SOFTWARE LIBRARY
[execution control]
SOFTWARE SERVICE EXECUTION CONTROLLER
15
16
[processing result]
RESULT PROCESSOR
[data processing instruction candidate]

FIG. 1

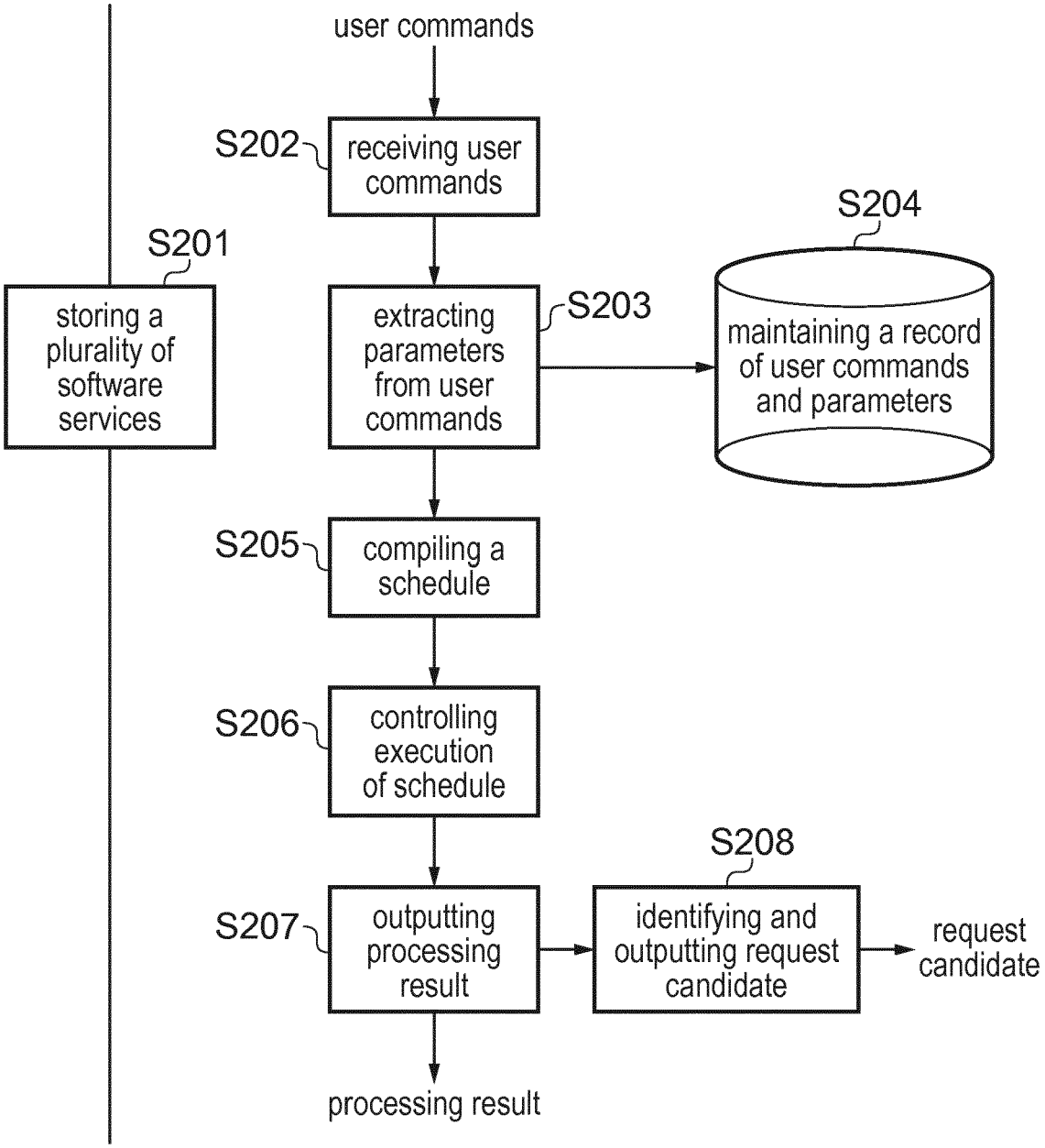
user commands
S202
receiving user commands
S201
storing a plurality of software services
S203
extracting parameters from user commands
S204
maintaining a record of user commands and parameters
S205
compiling a schedule
S206
controlling execution of schedule
S207
outputting processing result
S208
identifying and outputting request candidate
request candidate
processing result

FIG. 2

#	ID	Text
1	User:	Hi Aida
2	AIDA:	Hello Dave, how may I help you?
3	User:	Tell me, what is your name?
4	AIDA:	I am Aida v0.2
5	User:	Explore `http://example.com/file.csv as Dataset1`
6	AIDA:	The URL seems to point to a CSV file. Summary: Size: 2MB Rows: 200 Columns: 10 Row 1 seems to be a timestamp, run time series analysis? yes/no
7	User:	Yes
8	AIDA:	Dataset1 contains information from 2008/1/1 to 2015/12/31, each row has a unique timestamp Search for correlations between remaining columns? yes/no
9	User:	Yes
10	AIDA:	Column3 (labeled `Wind_Speed') and Column10 (labeled `Env_NO2') seem to be inversely correlated Train regression model? yes/no
11	User:	No
12	AIDA:	What should I do?
13	User:	store Dataset1
14	AIDA:	Operation successful. What next?
15	User:	run full analysis, generate report, and mail it
16	AIDA:	Assuming Dataset1 as last recently used variable.\label{line:omitted Task running in background... Awaiting orders.
17	User:	bye Aida
18	AIDA:	Ending session, have a nice day!

FIG. 3

EP 3 312 722 A1

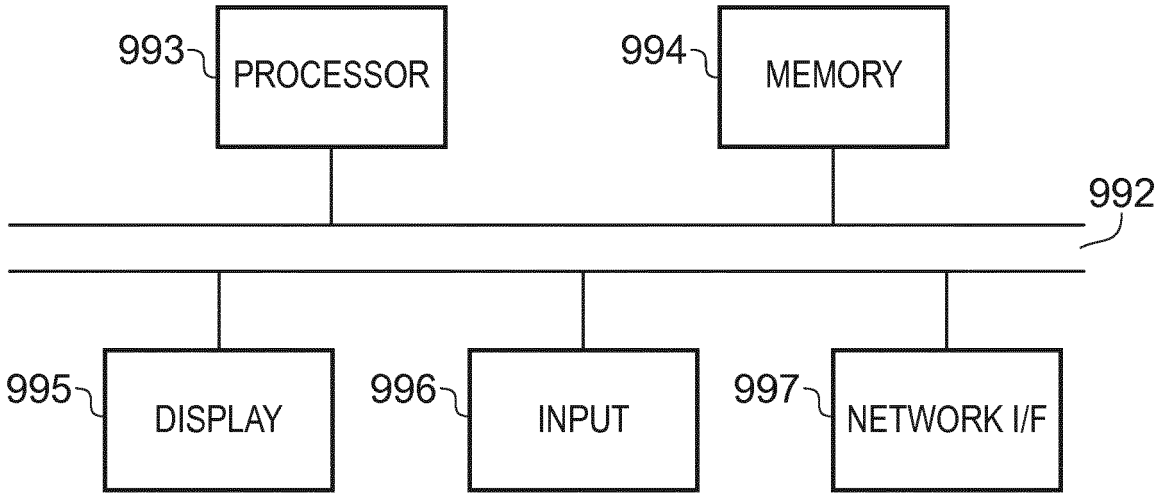
993
PROCESSOR
994
MEMORY
992
995
DISPLAY
996
INPUT
997
NETWORK I/F

FIG. 4

Europäisches Patentamt
European Patent Office
Office européen des brevets

EUROPEAN SEARCH REPORT

Application Number
EP 17 19 5347

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
X	WO 2016/066035 A1 (BAIDU ONLINE NETWORK TECHNOLOGY BEIJING CO LTD [CN]) 6 May 2016 (2016-05-06) * the whole document *	1-13	INV. G06F9/48 G06F9/50
X,P	& EP 3 096 226 A1 (BAIDU ONLINE NETWORK TECHNOLOGY BEIJING CO LTD [CN]) 23 November 2016 (2016-11-23) * paragraphs [0011], [0014], [0016] - [0019], [0030], [0032], [0035], [0040] - [0042], [0047], [0121]; claims 1,2,6; figure 1 *	1-13	
X	WO 2011/088053 A2 (APPLE INC [US]; GRUBER THOMAS ROBERT [US]; CHEYER ADAM JOHN [US]; KITT) 21 July 2011 (2011-07-21) * figures 1,2,7-11,23,27,31-33,37,47 *	1-4,12, 13	

TECHNICAL FIELDS SEARCHED (IPC)

G06F
G06N

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
The Hague	1 March 2018	Manfrin, Max

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 17 19 5347

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

01-03-2018

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2016066035 A1	06-05-2016	CN 104360897 A	18-02-2015
		EP 3096226 A1	23-11-2016
		JP 2017517776 A	29-06-2017
		KR 20160124766 A	28-10-2016
		US 2017242843 A1	24-08-2017
		WO 2016066035 A1	06-05-2016
WO 2011088053 A2	21-07-2011	AU 2011205426 A1	23-08-2012
		CA 2787351 A1	21-07-2011
		CA 2791791 A1	21-07-2011
		CA 2792412 A1	21-07-2011
		CA 2792442 A1	21-07-2011
		CA 2792570 A1	21-07-2011
		CA 2793002 A1	21-07-2011
		CA 2793118 A1	21-07-2011
		CA 2793248 A1	21-07-2011
		CA 2793741 A1	21-07-2011
		CA 2793743 A1	21-07-2011
		CA 2954559 A1	21-07-2011
		CN 102792320 A	21-11-2012
		CN 105808200 A	27-07-2016
		EP 2526511 A2	28-11-2012
		EP 3131023 A1	15-02-2017
		GB 2490444 A	31-10-2012
		JP 5948372 B2	06-07-2016
		JP 5956511 B2	27-07-2016
		JP 5957038 B2	27-07-2016
		JP 5973500 B2	23-08-2016
		JP 6027052 B2	16-11-2016
		JP 6175413 B2	02-08-2017
		JP 6193181 B2	06-09-2017
		JP 2013517566 A	16-05-2013
		JP 2014222509 A	27-11-2014
		JP 2014222510 A	27-11-2014
		JP 2014222511 A	27-11-2014
		JP 2014222512 A	27-11-2014
		JP 2014222513 A	27-11-2014
		JP 2014222514 A	27-11-2014
		JP 2014222515 A	27-11-2014
		JP 2014222516 A	27-11-2014
		JP 2014222517 A	27-11-2014
		JP 2017224300 A	21-12-2017
		KR 20120120316 A	01-11-2012
		KR 20120136417 A	18-12-2012
		KR 20120137424 A	20-12-2012
		KR 20120137425 A	20-12-2012

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 17 19 5347

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

01-03-2018

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
		KR 20120137434 A	20-12-2012
		KR 20120137435 A	20-12-2012
		KR 20120137440 A	20-12-2012
		KR 20120138826 A	26-12-2012
		KR 20120138827 A	26-12-2012
		KR 20130000423 A	02-01-2013
		KR 20160105995 A	08-09-2016
		KR 20170104006 A	13-09-2017
		MX 338784 B	02-05-2016
		MX 342072 B	13-09-2016
		MX 348250 B	05-06-2017
		RU 2012135502 A	27-02-2014
		RU 2012144605 A	27-04-2014
		RU 2012144606 A	10-05-2014
		RU 2012144637 A	10-05-2014
		RU 2012144639 A	10-05-2014
		RU 2012144640 A	10-05-2014
		RU 2012144643 A	10-05-2014
		RU 2012144644 A	10-05-2014
		RU 2012144647 A	10-05-2014
		RU 2012144648 A	10-05-2014
		RU 2015120954 A	27-12-2016
		US 2012016678 A1	19-01-2012
		US 2012245944 A1	27-09-2012
		US 2013110505 A1	02-05-2013
		US 2013110515 A1	02-05-2013
		US 2013110518 A1	02-05-2013
		US 2013110519 A1	02-05-2013
		US 2013110520 A1	02-05-2013
		US 2013111348 A1	02-05-2013
		US 2013111487 A1	02-05-2013
		US 2013117022 A1	09-05-2013
		US 2013185074 A1	18-07-2013
		US 2013185081 A1	18-07-2013
		US 2017178626 A1	22-06-2017
		WO 2011088053 A2	21-07-2011

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82